

Grounding Oriented Design

Christopher Stanton



A Thesis submitted for the degree of Doctor of Philosophy

Faculty of Information Technology

University of Technology, Sydney

2007

Certificate of Authorship and Originality

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student

Chu Mal

Acknowledgements

A huge “thankyou” goes to my supervisor, Professor Mary-Anne Williams. In late 2001, I was a disillusioned software engineer, bored with my chosen career path, drifting from one job to the next, searching for something more, and for some strange reason, I found myself spending my time at work daydreaming about artificial intelligence (perhaps because I wanted the machines I was programming to do the work for me). So, I made inquiries at my local university (Newcastle) about the possibility of undertaking a research honours degree in artificial intelligence. These inquiries led me to Mary-Anne, and as the cliché goes, the rest is history. Mary-Anne introduced me to robotic soccer, offered me the opportunity to undertake a PhD at UTS, entrusted with me an important leadership role (leading the software development for the UTS RoboCup team), allowed me the freedom to pursue the research areas that interested me, provided me with the opportunity to discuss the grounding problem with great minds such as John McCarthy and Peter Gärdenfors, and gave me the guidance to keep me pointed in the right direction. Not only was her guidance and support invaluable, but she has been a great mentor.

Another huge “thankyou” goes to my parents, Dr John Stanton and Dr Patricia Stanton. My parents, both academics, have obviously had an enormous influence on my being the person I am today. Their support, advice, proofreading and love, has been offered, as always, unconditionally.

Table of Contents

Table of Contents	iii
Abstract	xii
1 Introduction	1
1.1 Motivation	1
1.2 The Research Problem: Grounding	3
1.2.1 Grounding for Practical Robotics	4
1.2.2 Why is grounding important?	5
1.3 The Need for a Methodology	6
1.4 Scope	7
1.5 Thesis Objectives	8
1.6 Contribution	8
1.7 Scientific Method	9
1.8 Thesis Outline	10
2 Grounding: Approaches and Research Areas	12
2.1 Grounding: What is it?	12
2.1.1 Layman's grounding	13
2.1.2 Searle's Chinese Room	13
2.1.3 Harnad's Symbol Grounding	14
2.1.4 Brooks' Physical Grounding	15
2.1.5 Representation Grounding	15
2.1.5.1 Analog Computation	16
2.1.6 Autonomous Grounding	17
2.1.7 Summary	17
2.2 Meaning	18
2.2.1 Grounding and Meaning	18
2.2.1.1 Internalism vs Externalism	18
2.2.2 Intrinsic Semantics	19
2.2.3 A Theory of Reference	19
2.2.4 More than a theory of reference?	20
2.2.5 Summary	21
2.3 Approaches to Grounding	21
2.3.1 Harnad's Approach	22
2.3.1.1 Neural Networks	23

2.3.2	Categorical Perception	23
2.3.2.1	Machine Learning	24
2.3.2.2	Self-Organising Maps	24
2.3.2.3	Genetic/Evolutionary Algorithms	24
2.3.2.4	Conceptual Spaces	25
2.3.3	Symbolic Theft and Sensorimotor Toil	25
2.3.4	Hybrid Systems	27
2.3.5	Cognitivism versus Behaviourism	27
2.3.6	Developmental, Learning Systems	28
2.3.7	Summary	29
2.4	Grounding Research Areas	29
2.4.1	Language	29
2.4.1.1	Natural Language Processing	30
2.4.1.2	Computational Language Acquisition and Evolution	31
2.4.1.3	Psychological Experiments	33
2.4.1.4	Establishing "Common Ground" during Communication	33
2.4.2	"Dictionary Grounding"	34
2.4.3	Anchoring	34
2.4.4	Vision	35
2.4.4.1	Pastra's Double Grounding	35
2.4.4.2	Image Retrieval and the Semantic Gap	36
2.4.5	Action and Behaviour	36
2.5	Summary and Conclusion	37
3	Grounding: A Programmer's Perspective	38
3.1	A Working Definition	39
3.1.1	The Process of Embedding	40
3.1.2	Groundedness	40
3.2	The Difficulties of Grounding	41
3.2.1	Representation Design	42
3.2.1.1	The Relevance Problem	42
3.2.1.2	The Reference Problem	43
3.2.2	"Traditional" Software vs Robotics Software	43
3.2.2.1	Perception	44
3.2.2.2	Action	45
3.2.2.3	Requirements and Specification	46
3.2.2.4	Testing and Debugging	47
3.2.3	Summary	47
3.3	Towards a General Grounding Solution	48
3.3.1	Software Development Methodologies	49
3.3.2	Agent Oriented Software Engineering	49
3.3.3	Robotics Development Methodologies	50
3.3.3.1	Brooks' Subsumption Architecture	50
3.3.3.2	Wasson's Representation Design Methodology	51
3.3.3.3	Real-time Control Systems Architecture(RCS)	54
3.3.3.4	Behaviour Oriented Design	56
3.3.3.5	Roy's Grounding Framework	57
3.3.3.6	XABSL	59
3.4	Summary and Conclusion	59

4	Grounding Oriented Design:	
	Introduction and Overview	61
4.1	Motivation and Objectives	61
4.2	Methodology Scope	62
4.3	Design Considerations	62
4.3.1	Grounded Designers	63
4.3.2	A Software Problem	63
4.3.3	The Relevance Problem	63
4.3.4	Problem Decomposition and Decision-Making	63
4.3.5	The Reference Problem	64
4.3.6	Groundedness	64
4.3.7	Design Considerations - Summary	64
4.4	Methodology Overview	64
4.5	Summary	66
5	Grounding Oriented Design:	
	Context-Level Analysis	67
5.1	The Context-Level Skill	67
5.2	Objectives	69
5.3	Constraints	69
5.4	Current Capabilities	70
5.4.1	The Robot(s)	70
5.4.2	Software	71
5.5	Required Capabilities	72
5.5.1	Separating Requirements and Design	72
5.5.2	Eliciting Requirements - The Requirements Checklist	74
5.5.3	Requirement Templates	77
5.6	Conclusion	79
6	Grounding Oriented Design:	
	Part I - Basic-Design	80
6.1	Introduction	80
6.2	Skills	81
6.3	Skill Collaboration	81
6.4	Skill Diagrams	82
6.4.1	Design Considerations	82
6.4.2	The Basics	82
6.4.2.1	Skill Naming	84
6.4.2.2	Numeric Identifiers	84
6.4.2.3	The Decomposition Box	85
6.4.2.4	Flow-Of-Control	85
6.5	The Design Process	85
6.5.1	The Context-Level Skill	86
6.5.1.1	Notation: Context-Level Skill	87
6.5.2	Iterative Skill Decomposition	87
6.5.3	Identifying Skills	87
6.5.3.1	Example	88
6.5.4	Skill Templates	88
6.5.4.1	Example	92

6.5.5	Identifying Skill-Transitions	93
6.5.5.1	What-If Analysis	93
6.5.5.2	What-If Analysis Checklist	93
6.5.5.3	Examples	94
6.5.5.4	Skill-Transitions - Summary	97
6.5.6	Design: Keeping it Simple	98
6.5.6.1	Layering	98
6.5.6.2	Amalgamation	99
6.6	Summary	103
7	Grounding Oriented Design:	
	Part II - Detailed-Design	104
7.1	Detailed-Design	104
7.2	Skill Types	105
7.2.1	Decisions	105
7.2.2	Actions	105
7.2.3	Perceptions	106
7.2.3.1	Sensations versus Perceptions	106
7.2.4	Behaviours	106
7.3	Knowledge Representation	107
7.4	Skill Diagrams - Detailed-Design	107
7.4.1	Flows	107
7.4.2	Skill Types	108
7.4.3	Concepts, Percepts and Memories	108
7.5	The Design Process	108
7.5.1	Identify Skill Types	108
7.5.1.1	Guidelines for Identifying Skill Types	111
7.5.1.2	Example	113
7.5.2	Decompose Transition-Conditions	113
7.5.2.1	Example	114
7.5.3	Individual Skill Design	117
7.5.3.1	Skill Design: Decisions	117
7.5.3.2	Skill Design: Perceptions	118
7.5.3.3	Skill Design: Actions and Behaviours	121
7.5.4	Reviewing a Detailed Skill Architecture	122
7.5.4.1	Design Review Checklist	124
7.6	Summary	128
8	Discussion and Conclusion	129
8.1	Go-Design Evaluation	129
8.1.1	Benefits of Go-Design	129
8.1.2	Limitations of Go-Design	131
8.1.3	Comparative Assessment	132
8.1.3.1	Subsumption Architecture	132
8.1.3.2	BOD	133
8.1.3.3	Wasson's Representation Design Methodology	133
8.1.3.4	RCS	134
8.1.3.5	Roy's Grounding Framework	135
8.2	Future Work	135

8.2.1	Understanding Decomposition	135
8.2.2	Go-Design Development Environment	136
8.2.3	Evaluating Grounding Approaches	136
8.2.4	Escaping the Chinese Room	137
8.2.4.1	Understanding Meaning	137
8.2.4.2	Grounding through Prediction	138
8.3	Summary and Concluding Comments	138
A	Grounding Oriented Design:	
	The Step-by-Step Guide	141
A.1	Context-Level	141
A.1.1	Context-Level: Objectives	141
A.1.2	Context-Level: Constraints	141
A.1.3	Context-Level: Current Capabilities	142
A.1.3.1	The Robot(s)	142
A.1.3.2	Software	143
A.1.4	Context-Level: Required Capabilities	144
A.1.4.1	Requirement Templates	144
A.1.4.2	Requirements Elicitation Checklist	144
A.2	Basic-Design	144
A.2.1	Skills	146
A.2.2	The Basic-Design Process	146
A.2.3	The Context-Level Skill	148
A.2.4	Iterative Skill Decomposition	149
A.2.4.1	Identifying Skills	149
A.2.4.2	Identifying Skill Transitions	149
A.2.5	Skill Templates	150
A.2.6	Design: Keeping it Simple	151
A.3	Detailed-Design	151
A.3.1	Skill Types and Knowledge Representation	153
A.3.2	Identify Skill Types	153
A.3.2.1	Decisions	153
A.3.2.2	Actions	153
A.3.2.3	Perceptions	156
A.3.2.4	Behaviours	156
A.3.2.5	Guidelines for Identifying Skill Types	156
A.3.3	Decompose Transition-Conditions	158
A.3.4	Individual Skill Design	158
A.3.4.1	Skill Design: Decisions	158
A.3.4.2	Skill Design: Perceptions	159
A.3.4.3	Skill Design: Actions and Behaviours	160
A.3.5	Reviewing a Detailed Skill Architecture	160
A.3.5.1	Design Review Checklist	161
A.4	Detailed-Design Example Diagrams	162
	Bibliography	169

List of Figures

3.1	A representation of a possible subsumption architecture. Each layer represents an encapsulated behaviour (with its own sensing and acting), with higher levels possessing the ability to subsume lower-level behaviours.	51
3.2	A schematic diagram of a subsumption architecture robot capable of "hallway following" [20].	52
3.3	A decomposition diagram displaying a partial decomposition of the task "walk-the-dog" [136].	53
3.4	A flow diagram for the task "walk-the-dog" [136].	53
3.5	An example of an RCS architecture for controlling an autonomous vehicle. Boxes marked "SP" perform sensory processing, boxes marked "WM" perform world modeling, and boxes marked "BG" generate behaviours. Diagram taken from [4].	54
3.6	A "basic reactive plan" for a hungry monkey (p.30 [25]). Statements in brackets, e.g. "(have hunger)", are conditions upon which behaviours should be triggered. Thus, in this diagram if the monkey is hungry, it should get a banana, peel a banana, and then eat a banana.	56
3.7	A behaviour diagram from BOD (p.27) which represents a monkey that screeches depending upon who it recognises. Behaviours are represented by rectangular boxes and internal state is represented underneath the behaviour name.	57
3.8	An example of a "schema for a tangible (touchable, graspable, moveable, visible) object such as a cup", taken from Roy's Grounding Framework [108]. Legend: "Analog beliefs" are represented by ovals, "categorical beliefs" are represented by rectangles, "sensor projections" by triangles, and "action projections" by diamonds.	58
5.1	A context-level skill diagram for Play-Soccer	68
5.2	The raw data for an image (top) is quite meaningless to a human observer, whereas the corresponding data converted to an RGB image portrays a different picture. . .	73
5.3	Requirements for Movement	78

6.1	Skill Diagram Legend.	83
6.2	Shopping Skill Diagram.	84
6.3	The diagrammatic notation for representing concurrent skills. Move-Trolley and Find-Groceries operate concurrently.	86
6.4	A context-level skill diagram for a goal-kicking soccer robot.	87
6.5	A skill sequence for kicking a goal. The skill diagram lacks transition-conditions. . .	89
6.6	An alternate skill diagram for kicking a goal. Find-Ball is now a subskill of Get-Ball . The skill diagram lacks transition-conditions.	90
6.7	A skill template.	91
6.8	Skill template for Get-Ball	92
6.9	A skill sequence for kicking a goal with "successful" transition-conditions.	95
6.10	A skill diagram for Kick-Goal with the new "unsuccessful" transition-conditions unable-to-find-ball and unable-to-find-goal	95
6.11	A skill diagram for Kick-Goal after the addition of the "unsuccessful" skill-transitions ball-possession-lost and ball-lost	96
6.12	Kick-Goal , now with the ability to Get-Up after falling over.	97
6.13	Get-Up separated from Kick-Goal	98
6.14	Perceive-Posture	99
6.15	Kick-Ball-At-Goal - now without Get-Up	100
6.16	Simplified Kick-Ball-At-Goal (top) with Aim-At-Goal and Kick amalgamated into Aim-And-Kick (bottom).	101
6.17	Further simplified Kick-Ball-At-Goal (top) with Find-Ball and Get-Ball amalgamated (bottom).	102
7.1	Flow types in Go-Design.	108
7.2	Overview of skill types in Go-Design.	109
7.3	Knowledge Representation in Go-Design.	110
7.4	Skill types for Kick-Goal	113
7.5	Detailed-Design for Perceive-Posture	115
7.6	Pseudocode representing the decision-making for Perceive-Posture	116
7.7	Detailed skill diagram for Get-Up	116
7.8	Pseudocode representing the decision-making for Get-Up	117
7.9	Revised Kick-Goal to incorporate Perceive-Ball operating every processing cycle. .	119
7.10	Skill diagram for Perceive-Ball	120
7.11	Skill diagram for Spin-On-Spot in which there is no obstacle avoidance capability. .	122

7.12 Skill diagram for Spin-On-Spot in which the perception of obstacles and the behaviour to avoid obstacles is incorporated.	123
7.13 Simplified and improved skill diagram for Spin-On-Spot with obstacle avoidance capabilities. The Spin-On-Spot action is reused from Figure 7.11.	124
7.14 Corrected skill diagram for Spin-On-Spot-And-Avoid-Obstacle , which now chooses the spin direction for Spin-On-Spot . Figure 7.15 shows the corrected Spin-On-Spot . Note, the Ball percept is outside the decomposition box to represent the global nature of this percept.	126
7.15 Corrected Spin-On-Spot , which no longer chooses spin-direction. Rather, this memory is now set by Spin-On-Spot-And-Avoid-Obstacle , as displayed in Figure 7.14.	127
A.1 Go-Design Requirement Template.	145
A.2 Skill Diagram Legend.	147
A.3 Shopping Skill Diagram.	148
A.4 A context-level skill diagram for a goal-kicking soccer robot.	148
A.5 A skill template.	152
A.6 Overview of skill types in Go-Design.	154
A.7 Knowledge Representation in Go-Design.	155
A.8 Flow types in Go-Design.	155
A.9 Spin-On-Spot-And-Avoid-Obstacle . Note, the Ball percept is outside the decomposition box to represent the global nature of this percept.	163
A.10 Spin-On-Spot , implemented as an action, not a behaviour.	164
A.11 Detailed skill diagram for Get-Up	165
A.12 An implementation of the decision Check-For-Timeout . Note, frame-id is outside the decomposition-box to represent the global nature of this variable.	166
A.13 Detailed-Design for Perceive-Posture	167

List of Tables

A.1 Requirements Elicitation Checklist 168

Abstract

The symbol grounding problem[67] is a longstanding, poorly understood issue that has interested philosophers, computer scientists, and cognitive scientists alike. The grounding problem, in its various guises, refers to the task of creating *meaningful* representations for artificial agents. After more than 15 years of widespread debate and circular introspection of the so-called symbol grounding problem we seem none-the-wiser as to what constitutes being meaningful, and indeed grounded, for an agent[128].

We argue, in the context of practical robotics, a grounded agent possesses a representation which faithfully reflects pertinent aspects of the world. In contrast, an ungrounded agent could be, for example, delusional or suffering from hallucinations (“false positives”), overly concerned with irrelevant things (e.g. the frame problem[93]), or incapable of reliably perceiving, recognising or anticipating relevant things in a timely manner (“false negatives”). While most grounding research concerns how to develop agents which can autonomously develop their own representations (i.e. *autonomous* grounding), the fact all robotic systems are grounded through human design on a case-by-case, ad-hoc basis has been overlooked. This thesis presents *Grounding Oriented Design* - a methodology for designing and grounding the “minds” of robotic agents. Grounding Oriented Design (or, alternatively *Go-Design*) is a vital first step towards the development of autonomous grounding capabilities through improved understanding of the processes by which human designers ground robot minds.

Grounding Oriented Design offers guidelines and processes for iteratively decomposing a robot control problem into a set of collaborating skills, together with a notation for representing and documenting skill designs. Grounding Oriented Design consists of two main phases: basic-design which involves constructing a skill-architecture, and a detailed-design in which a skill-architecture is used to design the agent’s representation and decision-making processes. A groundedness framework[143] is used for describing and assessing the groundedness of either the complete system or of individual skills. Examples of the methodology’s use and benefits are provided, while suggestions for future work are discussed.